Google

Web Images Groups News Froogle Local more »

runtime lexical analyzer Search Advanced Search Preferences

Web Results **1** - **10** of about **65,400** for **runtime lexical analyzer**. (**0.20** seconds)

The Code Project - CLex: A Programmable **Lexical Analyser** - C++ / MFC
Create a **lexical analyzer** on the fly by constructing with a lex specification.
... This is achieved by the construction of a state machine at **runtime** from ...
www.codeproject.com/cpp/BHLex.asp - 49k - Sep 12, 2005 - Cached - Similar pages

PHP::sharp - A compiler for the .NET **runtime**
The phpLex utility is based upon the Lex **lexical analyzer** generator model. ...
If the %full directive is given, phpLex will generate a **lexical analyzer** that ...
php-sharp.sourceforge.net/index.php/6.html - 37k - Cached - Similar pages

Arcadia: CU Arcadia: Java Bison Parser **Runtime**
It is the primary **runtime** parse engine. It contains one class: yyparse. ...
It contains a subclass of yylex to do the actual **lexical analysis** appropriate to ...
serl.cs.colorado.edu/~arcadia/Software/jb.html - 9k - Cached - Similar pages

CMPSC 160 Course Description
Lexical analyzer generators, JLex **lexical analyzer** generator. ... **Runtime** environments [2 lectures, 1 discussion]. Symbol tables. ...
www.cs.ucsb.edu/~bultan/courses/160/ - 10k - Cached - Similar pages

MBLabSoft Company Site - Source Code Scanners 4.0
... set of regular expressions in a source code and are created at **runtime**. ...
TCppLexer is the component of the **lexical analyzer** of the C++ source codes; ...
www.mblabsoft.com/products/scs.html - 12k - Cached - Similar pages

JAVA-LEX & CUP Tutorial
Java-Lex is a **lexical analyzers** generator, while CUP is a LALR parsers ...
By definition, the parser tokens must be of java_cup.**runtime**.token type or a ...
www.hio.hen.nl/~vanleeuw/pse/spanje/tutorial.html - 16k - Cached - Similar pages

Computer Science - CS 321, 322 Languages and Compiler Design (4, 4 ...
Implement a **lexical analyzer**, parser, and type-checker for a simple but ...
Explain **runtime** organization of program execution, including activation records, ...
www.cs.pdx.edu/course.php?cid=6 - 11k - Cached - Similar pages

[PDF] Programming Assignment II Due Tuesday, February 26, 2002 ...
File Format: PDF/Adobe Acrobat - View as HTML
For this assignment you are to write a **lexical analyzer**, also called a. scanner.
, using a ... java cup.**runtime**.Symbol. as well as other supporting code ...
www.cis.udel.edu/~pollock/672/s02/a2.pdf - Sep 12, 2005 - Similar pages

Complier Construction Tools, Part III LG #41
The purpose of JFlex in this project is to build a **lexical analyzer** for our ...
The first class is for our parser, the next is the java_cup.**runtime**. ...
www.linuxgazette.com/issue41/lopes/lopes.html - 29k - Cached - Similar pages

Comp Review Spring 1996: Compilers
... When info about data is known only at **runtime** (eg. arrays in APL) - If number of

... Why separate **lexical analysis** phase? 1. Simpler lang design parser ...
www.personal.kent.edu/ ~rmuhamma/Compilers/compreview.html - 17k - Cached - Similar pages

Gooooooooooogle ▶
Result Page: 1 2 3 4 5 6 7 8 9 10 **Next**

Free! Instantly find your email, files, media and web history. Download now.

runtime lexical analyzer Search

Search within results | Language Tools | Search Tips | Dissatisfied? Help us improve

Google Home - Advertising Programs - Business Solutions - About Google

USPTO

Subscribe (Full Service) Register (Limited Service, Free) Login

Search: [✓ The ACM Digital Library] The Guide

runtime lexical analyzer

THE ACM DIGITAL LIBRARY

Feedback Report a problem Satisfaction survey

Terms used **runtime lexical analyzer** Found **2,621** of **160,906**

Sort results by: relevance

Display results: expanded form

Save results to a Binder

Search Tips

☐ Open results in a new window

Try an Advanced Search
Try this search in The ACM Guide

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next

Best 200 shown Relevance scale

1 Scalable instruction-level parallelism through tree-instructions
Jaime H. Moreno, Mayan Moudgil
July 1997 **Proceedings of the 11th international conference on Supercomputing**
Full text available: pdf(1.06 MB) Additional Information: full citation, references, index terms

2 TIL: a type-directed optimizing compiler for ML
D. Tarditi, G. Morrisett, P. Cheng, C. Stone, R. Harper, P. Lee
May 1996 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 1996 conference on Programming language design and implementation**, Volume 31 Issue 5
Full text available: pdf(1.23 MB) Additional Information: full citation, references, citings, index terms

3 LIMP: an interpreted programming language for students, professors and programmers
William Hawkins
January 2004 **Journal of Computing Sciences in Colleges**, Volume 19 Issue 3
Full text available: pdf(117.55 KB) Additional Information: full citation, abstract, references, index terms

Based upon an investigation of current interpreted and compiled programming languages, there exists a need for a new language with English-based keywords, looping constructs, functions, data type morphing and arrays. This paper describes a new language focused on these concepts: LIMP. To be useful, this new language is aimed towards users ranging from students to system administrators. LIMP was designed for ease of use and speed. This interpreted language was implemented in the fall of 2002 and ...

4 hcc—a portable ANSI C compiler (with a code generator for the PowerPCs)
Mohd Hanafiah Abdullah
August 1996 **ACM SIGPLAN Notices**, Volume 31 Issue 8
Full text available: pdf(447.88 KB) Additional Information: full citation, abstract, index terms

hcc is a portable optimizing compiler for ANSI C [8] [9] with code generators for the PowerPC and the MIMOS's RISC CPU. The PowerPC is a RISC processor developed jointly by IBM, Motorola, and Apple Computer among others in 1993, while the MIMOS's RISC CPU is an experimental 32-bit RISC processor. At the moment the compiler can be used to program PowerPC based computers including the PowerMacintoshes and the IBM RS6000 PowerPC based workstations.

5 Partial evaluation of general parsers
Christian Mossin
August 1993 **Proceedings of the 1993 ACM SIGPLAN symposium on Partial evaluation and semantics-based program manipulation**
Full text available: pdf(895.01 KB) Additional Information: full citation, abstract, references, citings, index terms

Applications of partial evaluation have so far mainly focused on generation of compilers from interpreters for programming languages. We partially evaluate a simple general LR(k) parsing algorithm. To obtain good results, we rewrite the algorithm using a number of binding-time improvements. The final LR(1) parser has been specialized using Similix, a partial evaluator for a higher order subset of Scheme [3]. The obtained specialized parsers are efficient and compact. ...

6 1996: TIL: a type-directed, optimizing compiler for ML
David Tarditi, Greg Morrisett, Perry Cheng, Chris Stone, Robert Harper, Peter Lee
April 2004 **ACM SIGPLAN Notices**, Volume 39 Issue 4
Full text available: pdf(2.02 MB) Additional Information: full citation, abstract, references

The goal of the TIL project was to explore the use of Typed Intermediate Languages to produce high-performance native code from Standard ML (SML). We believed that existing SML compilers were doing a good job of conventional functional language optimizations, as one might find in a LISP compiler, but that inadequate use was made of the rich type information present in the source language. Our goal was to show that we could get much greater performance by propagating type information through to t ...

7 Memory system performance of programs with intensive heap allocation
Amer Diwan, David Tarditi, Eliot Moss
August 1995 **ACM Transactions on Computer Systems (TOCS)**, Volume 13 Issue 3
Full text available: pdf(2.10 MB) Additional Information: full citation, abstract, references, citings, index terms

Heap allocation with copying garbage collection is a general storage management technique for programming languages. It is believed to have poor memory system performance. To investigate this, we conducted an in-depth study of the memory system performance of heap allocation for memory systems found on many machines. We studied the performance of mostly functional Standard ML programs which made heavy use of heap allocation. We found that most machines support heap allocation poorly. Howeve ...

Keywords: automatic storage reclamation, copying garbage collection, garbage collection, generational garbage collection, heap allocation, page mode, subblock placement, write through, write-back, write-buffer, write-miss policy, write-policy

8 Security and privacy: Securing web application code by static analysis and runtime protection
Yao-Wen Huang, Fang Yu, Christian Hang, Chung-Hung Tsai, Der-Tsai Lee, Sy-Yen Kuo
May 2004 **Proceedings of the 13th international conference on World Wide Web**
Full text available: pdf(2.67 MB) Additional Information: full citation, abstract, references, index terms

Security remains a major roadblock to universal acceptance of the Web for many kinds of transactions, especially since the recent sharp increase in remotely exploitable vulnerabilities have been attributed to Web application bugs. Many verification tools are discovering previously unknown vulnerabilities in legacy C programs, raising hopes that the same success can be achieved with Web applications. In this paper, we describe a sound and holistic approach to ensuring Web application security. Vi ...

Keywords: information flow, noninterference, program security, security vulnerabilities, type systems, verification, web application security

9 Memory subsystem performance of programs using copying garbage collection
Amer Diwan, David Tarditi, Eliot Moss
February 1994 **Proceedings of the 21st ACM SIGPLAN-SIGACT symposium on Principles of programming languages**
Full text available: pdf(1.28 MB) Additional Information: full citation, abstract, references, citings, index terms, review

Heap allocation with copying garbage collection is believed to have poor memory subsystem performance. We conducted a study of the memory subsystem performance of heap allocation for memory subsystems found on many machines. We found that many machines support heap allocation poorly. However, with the appropriate memory subsystem organization, heap allocation can have good memory subsystem performance.

10 Industrial strength compiler construction with equations
Lutz H. Hamel
August 1992 **ACM SIGPLAN Notices**, Volume 27 Issue 8
Full text available: pdf(455.75 KB) Additional Information: full citation, abstract, index terms

Industrial strength compilers need not only be robust and efficient but also understandable, maintainable, and extensible. To make compilers more manageable there has been a strong trend towards the high-level specification of the different phases of the compilation process. However, the high-level specification of one phase - the semantic analysis - has not been very successful in terms of production quality standards. In this paper we propose the use of a language based on equations for the sp ...

11 Interactive blackbox debugging for concurrent languages
G. Goldszmidt, S. Katz, S. Yemini
November 1988 **ACM SIGPLAN Notices , Proceedings of the 1988 ACM SIGPLAN and SIGOPS workshop on Parallel and distributed debugging**, Volume 24 Issue 1
Full text available: pdf(1.31 MB) Additional Information: full citation, abstract, references, citings, index terms

We describe a novel approach to the design of portable integrated debugging tools for concurrent languages. Our design partitions the tools set into two categories. The language specific tools take into account the particular features of a programming language for on-line experimenting with and monitoring of distributed programs. The language independent tools support off-line presentation and analysis of the monitored information. The separation of the lan ...

12 Papers: Arabic finite-state morphological analysis and generation
Kenneth R. Beesley
August 1996 **Proceedings of the 16th conference on Computational linguistics - Volume 1**
Full text available: pdf(452.87 KB) Additional Information: full citation, abstract, references, citings

This paper describes a large-scale system that performs morphological analysis and generation of on-line Arabic words represented in the standard orthography, whether fully voweled, partially voweled or unvoweled. Analyses display the root, pattern and all other affixes together with feature tags indicating part of speech, person, number, mood, voice, aspect, etc. The system is based on lexicons and rules from an earlier KIMMO-style two-level morpholgical system, reworked extensively using Xerox ...

13 Understanding the sources of variation in software inspections
Adam Porter, Harvey Siy, Audris Mockus, Lawrence Votta
January 1998 **ACM Transactions on Software Engineering and Methodology (TOSEM)**, Volume 7 Issue 1
Full text available: pdf(547.13 KB) Additional Information: full citation, abstract, references, citings, index terms

In a previous experiment, we determined how various changes in three structural elements of the software inspection process (team size and the number and sequencing of sessions) altered effectiveness and interval. Our results showed that such changes did not significantly influence the defect detection rate, but that certain combinations of changes dramatically increased the inspection interval. We also observed a large amount of unexplained variance in the data, indicating that other facto ...

Keywords: empirical studies, software inspection, software process, statistical models

14 Programming languages I: general purpose languages and compilers: Usage of an XPL based compiler generator system
Mark W. Storm, Jim A. Polk
April 1976 **Proceedings of the 14th annual Southeast regional conference**
Full text available: pdf(328.03 KB) Additional Information: full citation, abstract

This paper will discuss the expansion of an existing XPL compiler to include real arithmetic and formatted I/O. This was done at the University of Southern Mississippi on a Xerox Sigma 9 so that XPL can be used for numerical applications as well as giving the student user an introduction to the syntax of PL/I. The structure and use of the Compiler Generator will be traced through indicating what changes were made in the BNF grammar, the scanner, the semantic analyzer, and the runtime support fac ...

15 Saving traces for Ada debugging
Carol H. LeDoux, D. Stott Parker
May 1985 **ACM SIGAda Ada Letters , Proceedings of the 1985 annual ACM SIGAda international conference on Ada**, Volume V Issue 2
Full text available: pdf(732.37 KB) Additional Information: full citation, abstract, references, citings, index terms

A trace database model for debugging concurrent Ada programs is presented. In this approach, trace information is captured in an historical database and queried using Prolog. This model was used to build a prototype debugger, called Your Own Debugger for Ada (YODA). The design of YODA is described and a trace analysis of a sample program exhibiting misuse of shared data is presented. Because the trace database model is flexible and general, it can aid diagnosis of a variety of runtime errors.

16 Direct execution models of processor behavior and performance
Richard M. Fujimoto, William B. Campbell
December 1987 **Proceedings of the 19th conference on Winter simulation**
Full text available: pdf(952.49 KB) Additional Information: full citation, abstract, references, index terms

This paper discusses a modeling technique for creating efficient instruction level simulation models of von Neumann processors. In contrast to traditional approaches which use a software interpreter, this technique employs direct execution of application programs on the host computer. An assembly language program for the target machine is decompiled to a high level language, instrumented, and then recompiled and executed on the host computer. A prototype im ...

17 Fast detection of communication patterns in distributed executions
Thomas Kunz, Michiel F. H. Seuren
November 1997 **Proceedings of the 1997 conference of the Centre for Advanced Studies on Collaborative research**
Full text available: pdf(4.21 MB) Additional Information: full citation, abstract, references, index terms

Understanding distributed applications is a tedious and difficult task. Visualizations based on process-time diagrams are often used to obtain a better understanding of the execution of the application. The visualization tool we use is Poet, an event tracer developed at the University of Waterloo. However, these diagrams are often very complex and do not provide the user with the desired overview of the application. In our experience, such tools display repeated occurrences of non-trivial commun ...

18 Reserve paper: Incremental construction of a lexical transducer for Korean
Hyuk-Chul Kwon, Lauri Karttunen
August 1994 **Proceedings of the 15th conference on Computational linguistics - Volume 2**
Full text available: pdf(427.96 KB) Additional Information: full citation, abstract, references, citings

The paper describes the construction of a lexical transducer for Korean that can be used for stemming and generation. The method contains two innovations: (1) two-level rules as well-formedness constraints in the initial phase; (2) the combination of intersection and composition of rule transducers in a deep cascade for the final result.

Keywords: Korean lexical transducer, morphotactics, ordered rules, two-level morphology

19 Converting java programs to use generic libraries
Alan Donovan, Adam Kiežun, Matthew S. Tschantz, Michael D. Ernst
October 2004 **ACM SIGPLAN Notices , Proceedings of the 19th annual ACM SIGPLAN Conference on Object-oriented programming, systems, languages, and applications**, Volume 39 Issue 10
Full text available: pdf(1.18 MB) Additional Information: full citation, abstract, references, index terms

Java 1.5 will include a type system (called JSR-14) that supports <i>parametric polymorphism</i>, or <i>generic</i> classes. This will bring many benefits to Java programmers, not least because current Java practice makes heavy use of logically-generic classes, including container classes.

Translation of Java source code into semantically equivalent JSR-14 source code requires two steps: parameterization (adding type parameters to class definitions) and instantiation (a ...

Keywords: JSR-14, Java 1.5, Java 5, generic types, instantiation types, parameterized types, parametric polymorphism, raw types, type inference

20 Experiments with a Hindi-to-English transfer-based MT system under a miserly data scenario
Alon Lavie, Stephan Vogel, Lori Levin, Erik Peterson, Katharina Probst, Ariadna Font Llitjós, Rachel Reynolds, Jaime Carbonell, Richard Cohen
June 2003 **ACM Transactions on Asian Language Information Processing (TALIP)**, Volume 2 Issue 2
Full text available: pdf(152.01 KB) Additional Information: full citation, abstract, references, index terms, review

We describe an experiment designed to evaluate the capabilities of our trainable transfer-based (Xfer) machine translation approach, as applied to the task of Hindi-to-English translation, and trained under an extremely limited data scenario. We compare the performance of the Xfer approach with two corpus-based approaches---Statistical MT (SMT) and Example-based MT (EBMT)---under the limited data scenario. The results indicate that the Xfer system significantly outperforms both EBMT and SMT in t ...

Keywords: Evaluation, Hindi, example-based machine translation, limited data resources, machine learning, multiengine machine translation, statistical translation, transfer rules

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next

Ref #	Hits	Search Query	DBs	Default Operator	Plurals	Time Stamp
S24	1	(707/1-6.ccls.) and (generat$4 near4 token$2) same run$time Rev all	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2005/09/14 13:46
S23	2	717/143 and (generat$4 near4 token$2) and run$time	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2005/09/14 13:46
S18	149	(707/1-6.ccls.) and (generat$4 near4 token$2) Scan all	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2005/09/14 13:46
S22	8	717/143 and (generat$4 near4 token$2) Rev all	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2005/09/14 13:45
S21	234	717/143 Scan	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2005/09/14 13:45
S20	39	(717/139-144.ccls.) and (generat$4 near4 token$2) Rev all	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2005/09/14 13:45
S6	117	707/1-6.ccls. and generat$4 near4 token$2 Scan all	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2005/09/14 13:43
S17	16	(lexical adj analyzer) same (dynamic or run$time) Rev all	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2005/09/01 16:40
S1	149	(code or code) same (analyze or parse) same token$2 Scan	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2005/09/01 16:39